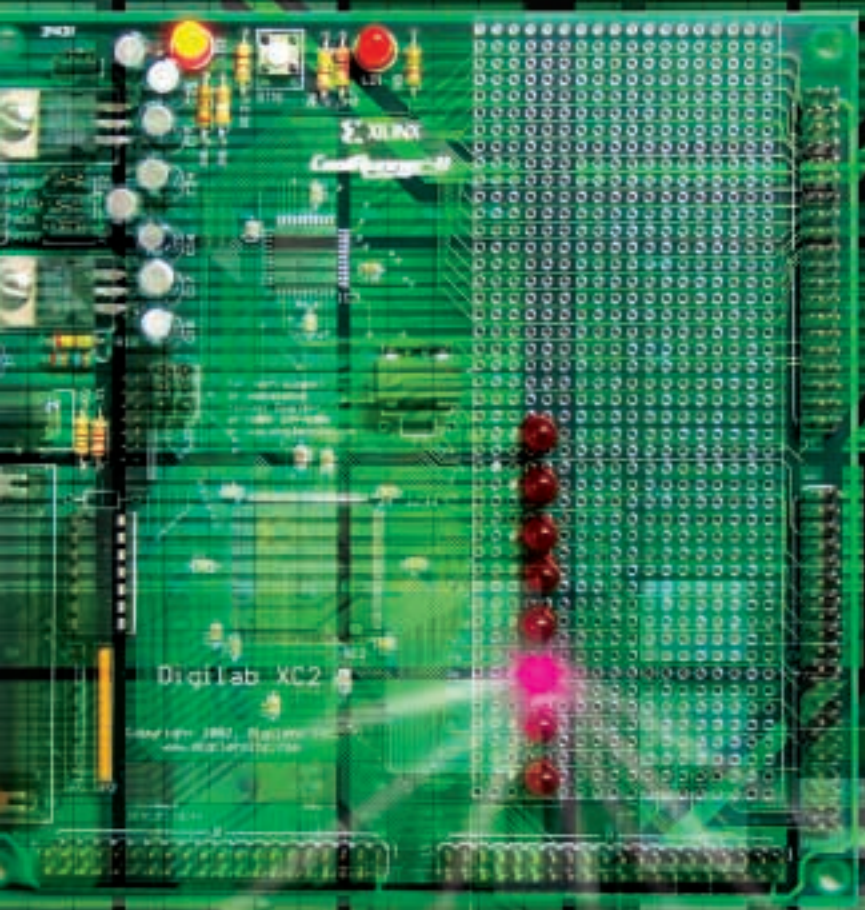
Reduce Your RISC with a PicoBlaze Reference Design

Build your own soft microcontroller in a CoolRunner-II CPLD. It's easy, fast, and free.



by Jesse Jenkins, Applications Manager
CPLD Business Unit
Xilinx, Inc.
jesse.jenkins@xilinx.com

Would you like to design your own microcontroller, but don't want all the hassle of starting from scratch with either the architecture or the support software?

Would you like to update older code from earlier microcontrollers to run faster and with the new memory standards?

Would you like to take the same code to substantially lower power dissipation?

Would you like to gain insight into the inner workings of a microcontroller with in-depth simulation as well as code creation support?

If the answer is yes to any of these questions, read on about the new Xilinx PicoBlaze™ microcontroller reference design. It's here now, it's fast, and maybe most important of all, it's free. This article details the CPLD version of the PicoBlaze reference design, and including where to get the application code, examples, and the cross assembler. With that under your belt, you are ready to start – but first, a little more detail.

PicoBlaze Explained

The PicoBlaze soft microcontroller is an 8-bit design that supports an 8-bit data bus and 16-bit instruction bus. As you may have guessed, the PicoBlaze design is based on the RISC (reduced instruction set computer) "Harvard architecture" model with separate data and instruction ports. The PicoBlaze design is written in VHDL, and is intentionally documented so that the accompanying cross assembler directly tracks the architecture.

The PicoBlaze version currently shipping supports 49 instructions that operate within any of several Xilinx CoolRunner™-II CPLDs. The speed will vary depending on exactly which instructions you wish to support and which version of the architecture you choose.

For instance, with the full instruction

set and all instructions held outside the CPLD, you can expect to achieve about 30 MHz performance. By streamlining either the instruction set or the program, you can triple performance to 90 MHz.

Naturally, the PicoBlaze microcontroller architecture takes advantage of the two key CoolRunner-II features – high-speed execution and low power consumption.

Add or Delete Instructions

Figure 1 shows the PicoBlaze base architecture, but don't restrict your thinking to that architecture alone. Think of it as a starting point. You are free to either add or delete capability as you see fit.

For instance, you can trim instructions from the instruction set by merely commenting them out of the VHDL. If you wish, you can also remove them from the assembler, but that is not required. You can also add instructions, if you have some application that can take advantage of essential instructions beyond those currently supplied.

It's also possible to do both – cut some instructions and add some instructions. For instance, most programmers use about 20 instructions in their day-to-day programming. Select the 20 you typically use, remove the rest, and then program. If you discover a bottlenecking "inner loop" that could benefit from a single instruction customized for that specific task, go ahead and write the VHDL that will do it at hardware speeds. Remember, the PicoBlaze microcontroller uses DualEDGE flip-flops within the processor to accomplish computation on both clock edges.

A DSP Example

To illustrate the ability of the PicoBlaze architecture to adapt, let's look at an example from DSP. The code to bit-reverse a bus is a fundamental operation used in Fast Fourier Transforms. The value is then typically driven out on the address lines as a critical step in the base algorithm. To do this in "standard" instructions would take multiple "mask and rotate" commands, creating a processing bottleneck.

Figure 2 shows the basic operations in assembler-like steps to display register contents. The algorithm starts with a byte of data labeled A-H. This byte is first internally swapped (four rotates), then successively, inner bits are picked off with Boolean AND/OR into a target register that will build up the results, two bits at a time. One pass through this results in the final register with the original contents reversed. Depending on algorithm details, it can take approximately 12 to 18 instructions. In this case, we dispense with adding the overhead of loop management with pointers and counters and unroll it.

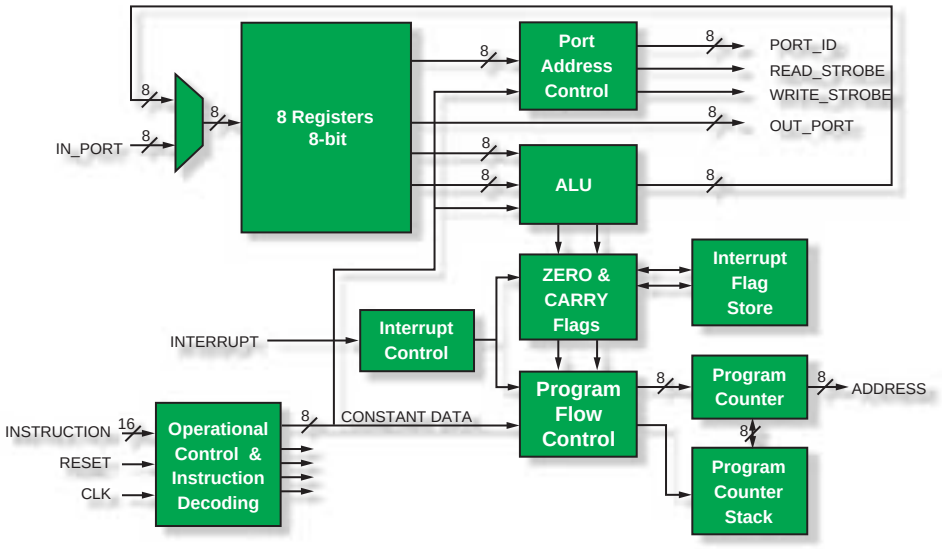


Figure 1 - PicoBlaze architecture

In Figure 3, the "flip" instruction is added to the VHDL, the design is recompiled, and the processor is "rewired" to add in this key instruction. This method collapses many instructions down to some gate rewiring with the synthesis tools. Very many bit-level operations boil down to simply rewiring the CPU, and best of all, the synthesizer does the work. Many other examples exist. See the links at the end of this article.

Figure 2 - Bit-reversal code steps

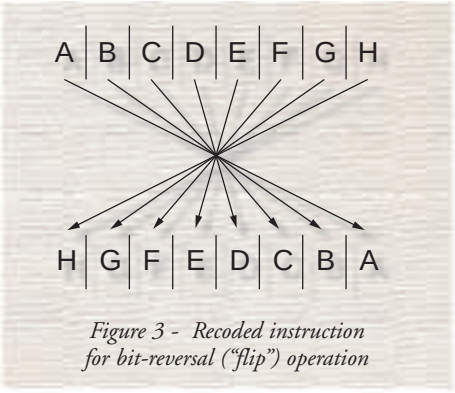


Figure 3 - Recoded instruction for bit-reversal ("flip") operation

Processor Enhancement

We just mentioned instruction set optimization, but it's also possible to add functionality. Remember that many microcontrollers include on-board function blocks that have a payoff beyond the instruction set. For instance, many 8-bit microcontrollers include internal peripheral counters or timers, interrupt handlers, and DMA circuits. With PicoBlaze, just add the right set of peripheral capability within the chip, depending on the density of the CoolRunner-II CPLD chosen. Table 1 shows the densities available in CoolRunner-II CPLDs, and Table 2 gives some estimates of macrocell usage for various add-on functions.

One very important thing to remember is that when choosing a function to add in, select just the functionality actually needed, so you will get the best usage from your choice. Bill Carter, one of the founders of Xilinx, frequently comments that most people don't really want or need a UART (universal asynchronous receiver/transmitter), but only an "RT." That is, they select a function that comes with 50 options, then only use two. They end up carrying along lots of unused circuitry that they pay for but never really use. Don't fall into that trap. Select the functions you will need and get the cheapest, fastest, lowest power solution possible. Note that most of the items listed in Table 2 exist as separate reference designs and may be found on the Xilinx website.

CoolRunner-II	XC2C32	XC2C64	XC2C128	XC2C256	XC2C384	XC2C512
Macrocells	32	64	128	256	384	512
MAX I/O	33	64	100	184	240	270
TPD (ns)	3.5	4.0	4.5	5.0	5.5	6.0
TSU (ns)	1.7	2.0	2.1	2.2	2.3	2.4
TCO (ns)	2.8	3.0	3.4	3.8	4.2	4.6
Fsystem1 (MHz)	333	270	263	238	217	217

Table 1 - CoolRunner-II macrocell capacities and pertinent data

Performance Improvement

Getting the most out of your design will be another step. A classic way to improve the design is to "tune" it. Observe its performance behavior, identify where the processor is spending its time, discover what it is doing, and think through the best set of operations to improve. Then, implement a new version of the architecture and/or code and evaluate it again.

One easy way to do that is with the CoolRunner-II Design Kit (see Figure 4). Many target designs easily fit onto the 256-macrocell XC2C256 that resides on that board. There is also a blank pinout site for adding a 64 macrocell XC2C64, with signals already attached to the XC256. Simply construct a small hardware performance monitor that will time various code sections and report back the execution time. That way, by examining the behavior over address space and time, you can determine just how much time is spent doing the various tasks.

Figure 5 shows a simple approach to doing this operation. With care, both

Function	Macrocells
IrDA and UAR/T	87
Timer/Counter	16 and up
DMA Port	16-32
Manchester Encoder/Decoder	55
Wireless XCVR	156
16b/20b Encoder/Decoder	76
Flash NAND Interface	9
SPI Interface	135
UAR/T	61
DDR SDRAM Interface	128
SMBus Controller	158

Table 2 - Common functions and approximate CoolRunner-II macrocell usage

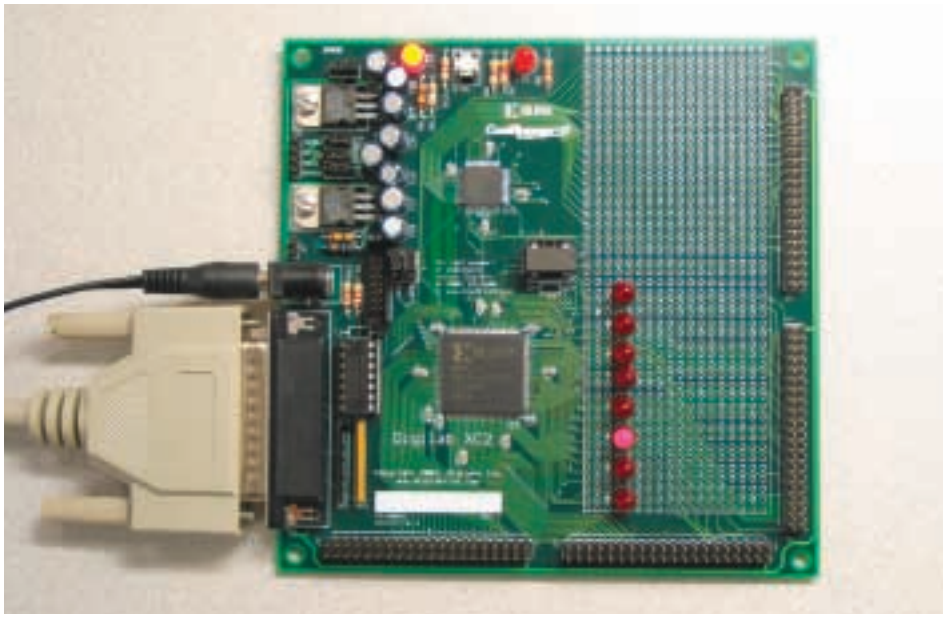


Figure 4 - CoolRunner-II Design Kit

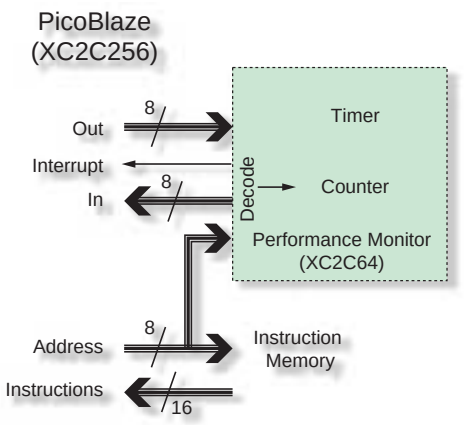


Figure 5 - PicoBlaze performance monitor

CPLDs can communicate through a PC parallel port via their JTAG boundary scan chains. Performance monitoring can help decide which aspects of the PicoBlaze design to perform in software and which to embody in hardware. One beautiful thing about building functions out of programmable logic is that different experiments to focus on specific performance targets are easily developed – giving you highly tuned, fast designs.

And you don't have to worry about power enhancement. CoolRunner-II CPLDs are already the lowest power CPLDs available today, and PicoBlaze is a very competitive low power microcontroller.

PicoBlaze Cross Assembler

As mentioned before, the PicoBlaze cross assembler is so well documented that a direct correspondence between assembly code and VHDL in the PicoBlaze design file already exists. The translator is written in ANSI-C and is assembled on Microsoft assemblers. The cross assembler is highly transportable and supports multiple output file types. For instance, it produces a binary output file ready to load into external EPROM in Intel hex format.

The cross assembler also produces the essential modeling files for the VHDL simulator. You can instantly analyze your produced code with high-speed simulations to determine the functionality and effectiveness of the code you have implemented. Then download the code into the CoolRunner-II Design Kit and see that they actually do work as expected.

Conclusions and Recommendations

This introduction to designing PicoBlaze microcontrollers was written to stimulate your imagination to discover the fascinating world of creating your own CPUs. Once you start, it can be addictive. You can easily alter the processor to be 16 or even 32 bits wide – or even non-binary powers. Then you will discover which instructions burn through the macrocells and what the new speed limits will be.

Do you need some instructions for 8 bits and others for 16? It's up to you. Application areas where these kinds of processors make sense include industrial control, low-power portable DSP (brainwaves, EKG, medical), and cryptography. Did you know that most cryptographic operations are bit-level operations and almost never do floating point arithmetic?

The PicoBlaze microcontroller reference design for CPLDs has been built, tested, and is now available over the Internet, free to the user.

Acknowledgements

Xilinx engineer Ken Chapman, who received additional support and encouragement from Henk van Kampen at Mediatronix BV, developed the original reference design. The CPLD version was created by Scott Lien, who also wrote the PicoBlaze cross assembler and the application note that is on the Xilinx website. The VHDL and cross assembler (source and executable) links are shown in xapp387 (see below).

For more information, see the following:

CoolRunner-II Design Kit:
www.xilinx.com/products/cpldsolutions/demoboard.htm (purchasing details)

CoolRunner-II Application Notes
www.xilinx.com/xapp/xapp375.pdf (timing model)
www.xilinx.com/xapp/xapp376.pdf (logic engine)
www.xilinx.com/xapp/xapp377.pdf (low-power design)
www.xilinx.com/xapp/xapp378.pdf (advanced features)
www.xilinx.com/xapp/xapp379.pdf (high-speed design)
www.xilinx.com/xapp/xapp380.pdf (cross point switch)
www.xilinx.com/xapp/xapp381.pdf (demo board)
www.xilinx.com/xapp/xapp382.pdf (I/O characteristics)
www.xilinx.com/xapp/xapp383.pdf (single error correction, double error detection)
www.xilinx.com/xapp/xapp384.pdf (DDR SDRAM interface)
www.xilinx.com/xapp/xapp387.pdf (PicoBlaze microcontroller)
www.xilinx.com/xapp/xapp388.pdf (on-the-fly reconfiguration)
www.xilinx.com/xapp/xapp389.pdf (powering CoolRunner-II CPLDs)

CoolRunner-II Data Sheets
www.xilinx.com/bvdocs/publications/ds090.pdf (CoolRunner-II CPLD family data sheet)
www.xilinx.com/bvdocs/publications/ds091.pdf (XC2C32 data sheet)
www.xilinx.com/bvdocs/publications/ds092.pdf (XC2C64 data sheet)
www.xilinx.com/bvdocs/publications/ds093.pdf (XC2C128 data sheet)
www.xilinx.com/bvdocs/publications/ds094.pdf (XC2C256 data sheet)
www.xilinx.com/bvdocs/publications/ds095.pdf (XC2C384 data sheet)
www.xilinx.com/bvdocs/publications/ds096.pdf (XC2C512 data sheet)

CoolRunner-II White Papers
www.xilinx.com/publications/products/cool2/wp_pdf/wp165.pdf (chip scale packaging)
www.xilinx.com/publications/whitepapers/wp_pdf/wp170.pdf (security)